Data Structures I: Binary trees

Mauricio Toro
Department of Systems and Informatics
Universidad EAFIT

Cocktail of the day: Mojito

Disclaimer: Keep alcohol out of the hands of minors.

Cocktail of the day: Mojito

- 40 ml of white rhum
- 30 ml of lime juice
- 30 ml of soda water
- 6 leaves of mint
- 2 teaspoons of sugar

OCR uses trees

https://www.youtube.com/watch?v=tVQtjCZaOKU

Applet to simulate a Linked list

http://visualgo.net/list.html

Binary tree

A binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child.

1. Applications
2. Implementation
3. Some Algorithms
   - Number of elements
   - Maximum Height
   - Search
   - Recursive print

Applications

- Files systems
- Family trees
- Databases
- Priority Queues
- Syntax trees
- ...

Taken from `http://stackoverflow.com/questions/2130416/what-are-the-applications-of-binary-trees`

Applications

- Files systems
- Family trees
- Databases
- Priority Queues
- Syntax trees
- ...

Taken from `http://stackoverflow.com/questions/2130416/what-are-the-applications-of-binary-trees`

Applications

- Files systems
- Family trees
- Databases
- Priority Queues
- Syntax trees
- ...

Taken from `http://stackoverflow.com/questions/2130416/what-are-the-applications-of-binary-trees`

Applications

- Files systems
- Family trees
- Databases
- Priority Queues
- Syntax trees
- ...

Taken from `http://stackoverflow.com/questions/2130416/what-are-the-applications-of-binary-trees`

Applications

- Files systems
- Family trees
- Databases
- Priority Queues
- Syntax trees
- ...

Taken from http://stackoverflow.com/questions/2130416/what-are-the-applications-of-binary-trees

Applications

- Files systems
- Family trees
- Databases
- Priority Queues
- Syntax trees
- ...

Taken from `http://stackoverflow.com/questions/2130416/what-are-the-applications-of-binary-trees`

Node of a binary tree

```
public class Node {
 public Node left;
 public Node right;
 public int data;
 public Node(int d){
     data = d;
 }
}
```

Node of a binary tree

```
public class BinaryTree {
   Node root;
   public BinaryTree() {
      root = null;
   }
}
```

Number of elements of a Tree

```
public class BinaryTree {
   ...
   private int elementsAUX(Node node) {
      if (node == null)
          return 0;
      else
          return elementsAUX(node.left)+
                 elementsAUX(node.right)+1;
   }
   public int elements() {
       return elementsAUX(root);
   }
}
```

Maximum Height of a Tree

```
public class BinaryTree {
  private int maxheightAUX(Node node) {
    if (node == null)
        return 0;
    else
        return Math.max(
          maxheightAUX(node.left),
          maxheightAUX(node.right))+1;
  }
  public int maxheight() {
     return maxheightAUX(root);
  }
}
```

Depth-first traversal

```
public class BinaryTree {
   ...
   private void printAUX(Node node) {
      if (node != null) {
            System.out.println(node.data);
            printAUX(node.left);
            printAUX(node.right);
      }
   }
   public boolean print() {
    return printAUX(root);
   }
}
```

DFS Complexity

```
public class BinaryTree {
   ...
   private void printAUX(Node node) {
      if (node != null) {
         System.out.println(node.data); //O(1)
         printAUX(node.left); //O(n/2)
         printAUX(node.right); //O(n/2)
}}}
```

- $T(n) = 2T(n/2) + C$
- $T(n)$ is $O(n)$

DFS Complexity

```
public class BinaryTree {
   ...
   private void printAUX(Node node) {
      if (node != null) {
         System.out.println(node.data); //O(1)
         printAUX(node.left); //O(n/2)
         printAUX(node.right); //O(n/2)
}}}
```

- $T(n) = 2T(n/2) + C$
- $T(n)$ is $O(n)$

Depth-first search

```
public class BinaryTree {
  private boolean findAUX(Node node, int x){
    if (node == null)
      return false;
    else
        return node.data == x ||
        findAUX(node.left,x) ||
        findAUX(node.right,x);
  }
  public boolean find(int x) {
   return findAUX(root, x);
  }
}
```

# Breadth-first traversal
import java.util.LinkedList;

```
public class BinaryTree {
  private void printByLevels() {
    LinkedList<Node> queue = new LinkedList();
    if (root != null) queue.add(root);
    while (queue.size() != 0) {
        Node node = queue.pop();
        System.out.println(node.data);
        if (node.left != null)
            queue.add(node.left);
        if (node.right != null)
            queue.add(node.right);
}}}
```

References

- Please learn how to reference images, trademarks, videos and fragments of code.
- Avoid plagiarism

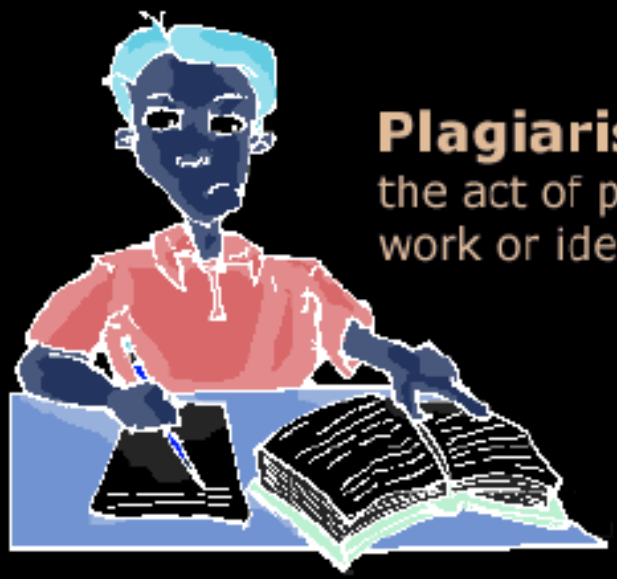

Figure: Figure about plagiarism, University of Malta [Uni09]

University of Malta.

Plagarism — The act of presenting another's work or ideas as your own, 2009.

[Online; accessed 29-November-2013].

Further reading

- Binary trees
  - `https://en.wikipedia.org/wiki/Binary_tree`